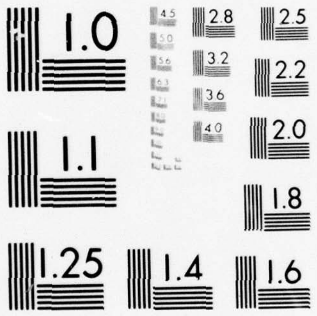
AD-A039 060 MITRE CORP BEDFORD MASS F/G 9/2
AUDIT AND SURVEILLANCE OF MULTI-LEVEL COMPUTING SYSTEMS.(U)
APR 77 C ENGELMAN F19628-75-C-0001
UNCLASSIFIED MTR-3207 ESD-TR-76-369 NL

| OF |
AD
A039060

END
DATE
FILMED
5-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3

12

ESD-TR-76-369 MTR-3207

AUDIT AND SURVEILLANCE OF MULTI-LEVEL COMPUTING SYSTEMS

APRIL 1977

Prepared for

DEPUTY FOR COMMAND AND MANAGEMENT SYSTEMS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Bedford, Massachusetts

Project No. 522N
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-75-C-0001

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

William R Price
WILLIAM R. PRICE, Captain, USAF
Project Engineer/Scientist

[illegible]
DONALD P. ERIKSEN
Project Engineer/Scientist

FOR THE COMMANDER

Frank J. Emma
FRANK J. EMMA, Colonel, USAF
Director, Computer Systems Engineering
Deputy for Command & Management Systems

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 **REPORT DOCUMENTATION PAGE**

READ INSTRUCTIONS BEFORE COMPLETING FORM

18 1. REPORT NUMBER ESD-TR-76-369	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 AUDIT AND SURVEILLANCE OF MULTI-LEVEL COMPUTING SYSTEMS.		5. TYPE OF REPORT & PERIOD COVERED
	14	6. PERFORMING ORG. REPORT NUMBER MTR-3207
7. AUTHOR(s) 10 C. Engelman	15	8. CONTRACT OR GRANT NUMBER(s) F19628-75-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS The MITRE Corporation Box 208 Bedford, MA 01730	12 41p.	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Project No. 522N
11. CONTROLLING OFFICE NAME AND ADDRESS Deputy for Command and Management Systems Electronic Systems Division, AFSC Hanscom Air Force Base, Bedford, MA 01731	11	12. REPORT DATE APR[redacted]977 13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 9 Technical rept.;		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

AUDIT
COMPUTER SECURITY
MULTI-LEVEL SECURITY
SURVEILLANCE

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Audit and surveillance are two techniques for observing the way that users employ a computer resource. Audit addresses an analysis of behavior after the fact, while surveillance is the active monitoring of user behavior for certain characteristics. This paper addresses audit and surveillance techniques as they augment the security controls of computer systems. Special attention is given to computer systems built

(over)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 235 050 UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (concluded)

around a security kernel. Many of the issues are discussed as they pertain to a secure Multics, a system chosen for its inherent ability to support security controls.

ACKNOWLEDGMENT

This report has been prepared by The MITRE Corporation under Project No. 522N. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

TABLE OF CONTENTS

TABLE OF CONTENTS (concluded)

SECTION I

INTRODUCTION

BACKGROUND

Most non-dedicated systems have security provisions that grant approved users access to the system, and then only for non-prohibited purposes within previously specified limits on the resources consumed. These security provisions are intended to ensure that no user can access private data or system data unless he has been granted sufficient access privileges by someone with proper authority. These provisions are further intended to protect the system integrity from attacks such as a user's attempt to deny services to others, to increase his own privileges, or decrease his own costs.

Most systems contain audit and surveillance subsystems intimately linked to these security and integrity functions. They serve generally to record data concerning such events as all successful (and, often, all unsuccessful) logins, all creations and most recent modifications of files, and all granting and rescinding of access privileges to the protected data files.

Somewhat aside from these security and integrity considerations, the performance of the system is usually heavily monitored, e.g., records are maintained of the occurrence of crashes, their nature (whether due to hardware or software), the percentage of CPU utilization, the on-line vs. batch consumption, and system reconfigurations (whether dynamic or off-line) in response to component failures. Such surveillance often serves a limited public function, as in the provision to all users of such information as the number of present users, the current system load, the time of the last crash, and the identity of the other users.

MULTI-LEVEL SECURE COMPUTING SYSTEMS

Before we go on to discuss security, and the audit and surveillance mechanisms that are appropriate in a multi-level secure environment, a use of audit and surveillance that is sometimes proposed should be immediately discredited. It has been suggested that audit and surveillance can be used to provide system security. It should be apparent, however, that if you know enough about access control to list all accesses (audit) or detect when an access is about to happen (surveillance), then you know how to prevent the access in the first place. Thus, the discussion that follows will address an environment in which a security kernel provides effective access controls, and audit and surveillance appropriately augment these controls.

Military security policy [1] regulates the handling of classified information. It requires and sets rules for the assignment of personal responsibility for document classification, the clearance of individuals for access privileges to information of varying security levels, the accounting of responsibility for individual documents and the modes and conditions under which such documents can be transferred. This policy further serves to give a formal meaning to the word "security" that can and must be translated into the context of computer systems that process data bearing military classification. The application of security policy to electronic data processing gains a new dimension of significance with the advent of multi-level secure systems, i.e., computer systems that support facilities for the simultaneous servicing of users with varying security clearances accessing shared data of varying security classification.

Examples of multi-level systems currently constructed or under construction are the Multics System at the Air Force Data Services Center [2], the Security Kernel Based Multics System, the Prototype Secure Front End Processor [3], the PDP-11/45 Security Kernel at The MITRE Corporation [4], and systems under development such as SATIN IV and some upgraded versions of the AABNCP. The extreme sensitivity of classified military data, combined with the tenable assumption of an espionage agent having large available resources, places much higher demands on the credibility and demonstrability of security in such systems than is normally placed on those "secure" systems constructed for civilian applications. These demands have led to the development of a technology, based on the concept of a reference monitor [5] that monitors all references to information, is tamperproof and can be proven correct.

The Multics system was recognized as a viable candidate to support multi-level security controls quite early in the Air Force program. In fact, an early evaluation of this machine by Anderson [6] contained an appendix that addresses the utility of audit in such a system. In the discussion that follows, many of the observations are directed towards the computational environment provided by Multics.

IMPACT ON AUDIT AND SURVEILLANCE

DoD 5200.28-M, "Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating - Secure Resource - Sharing ADP Systems", contains a section on audit that we reproduce in its entirety:

> SECTION V
> AUDIT LOG OR FILE
>
> 5-100 Application
>
> An audit log or file (manual, machine, or a combination of both) shall be maintained as a history of the use of the ADP System to permit a regular security review of system activity. (e.g., The log should record security related transactions, including each access to a classified file and the nature of the access, e.g., logins, production of accountable classified outputs, and creation of new classified files. Each classified file successfully accessed <regardless of the number of individual references> during each "job" or "interactive session" should also be recorded in the audit log. Much of the material in this log may also be required to assure that the system preserves information entrusted to it.)

This security manual is addressed to data processing systems in general, not just on-line multi-security-level systems and much of the above paragraph would be equally applicable to single-level systems. While the audit requirements outlined imply a minimum capability that must (and which, for many systems, can easily) be supplied, we must inquire whether multi-level security does not carry much broader implications for audit and surveillance.

While our response to the above question is certainly affirmative, we do not feel that it is possible to identify, at this time, the additional facilities required or those that are cost effective for multi-level systems. The primary purpose of this report is to outline some of the more prominent issues in audit and surveillance, and to identify a reasonable course of action to address these issues. Nonetheless, it does strike us that the general tone of the appropriate response will depend sharply on one critical distinction, viz, whether we are dealing with a system engineered for security or one whose security has been mathematically demonstrated.

Secure Systems in Controlled Environments

A secure system in a controlled environment is one designed to protect against compromise of information in a multi-level environment, but which has not been subject to a thorough mathematical validation. Such a system will be called a plausibly secure system. The Air Force Data Service Center System [2] is an example of such a system. For these systems, we imagine that a quite significant role would be played by surveillance for the purpose of detecting user behavior that might indicate a breach in security or an attempt at such a breach. Such surveillance might even lead to the timely arrest of an implicated partner or author of a malicious "Trojan horse" [7] procedure. These techniques may even extend into more active entrapment strategies for such arrest or, at least, the channeling of such attempts into innocuous pathways. Audit may be used for the estimation of the likelihood of prior damage due to successful penetration attempts, hopefully with some indication of those data areas most likely compromised.

Demonstrably Secure Systems

A demonstrably secure system has had its security controls demonstrated (mathematically proven) to be effective. In such a system, individual responsibility can be reliably enforced. In these systems we would not want to imply that all the surveillance considerations associated with non-verified systems disappear. There will always be someone who has sufficient reservations to suggest that it cannot hurt to apply such techniques, especially if they have already been developed. Furthermore, there may well remain a would-be penetrator who thinks he might be able to subvert the protection mechanisms and who tries his hand at it. We should be prepared to apprehend him.

However, there may turn out to be a novel use of an audit facility in conjunction with a demonstrably secure multi-level system in which the audit serves, in effect, not simply to record the functioning of the system, but, rather interestingly, to enhance it. This possibility stems from the methodology by which such certifiable systems as the PDP-11/45 kernel [4] or the Multics kernel are designed and validated. In these systems, the memory is a virtual one, divided into segments each of which bears a security level. All memory references are localized to and monitored by a security kernel that guarantees that security is preserved by the imposition of a sufficient condition, called the "*-property" [8]. The *-property provides the assurance that no process can write data at a lower classification than the maximum level it is entitled to read. Now, while the *-property provides an adequate basis for the construction of a demonstrably secure multi-level system, it may be possible to relax this sufficient access check. The check would be relaxed in that, if the *-property were to be violated on an access, then the access would be enabled, but audited. A permanent record would then be kept of all such *-property violations.

In essence, the user would be empowered to make the same decisions he is empowered to with respect to paper documents. The essential role of the audit is to provide a record of any advantage a "Trojan horse" might have taken of the relaxation of this access rule. This novel significance of this audit technique is discussed at greater length in Section II, under the title, "FUNCTIONAL AUDITING". Of course, audit provides the accountability for need-to-know controls.

PURPOSE AND OUTLINE OF THIS REPORT

The purpose of this report is to serve as a guideline for ESD in planning the audit and surveillance of multi-security-level computer systems. Special emphasis is placed on the secure computer systems being developed under ESD direction. We intend this report to be available as both an introduction to the issues involved in the audit and surveillance of such systems and as a partial checklist for the direction and monitoring of anyone developing such facilities.

Following this introduction, Section II discusses the purposes and goals of audit and surveillance of multi-level computer systems. Section III discusses the constraints that might be imposed on audit and surveillance strategies, e.g., cost constraints or conflicts with privacy or integrity considerations. Section IV lists a number of "strategies", i.e., approaches that might be taken both to the provision of audit and surveillance functions within multi-level secure computer systems and to the study program that might clarify those surveillance approaches that seem most sound and cost-effective. Finally, Section V presents a set of recommendations to ESD as to what we feel represents a reasonable course to proceed along with respect to the stucy, development, and implementation of audit and surveillance sub-systems of on-line multi-security-level computer systems. These range from the immediate provision of "statutory" audit facilities to the feasibility study and perhaps eventual installation of some of the more imaginative surveillance proposals.

SECTION II

GOALS

THE MECHANICS OF DATA CLASSIFICATION

To speak of the preservation of security in a computer system implies the presence of certain minimal facilities. In particular, each user must have a known security level, i.e., the maximum level (1) of the process he can invoke; each process must have a security level, i.e., the maximum security level of data it can read and the minimum it can write; and each data structure must have a security level. In practice, a good deal more is usually present. Each user, for example, not only has an associated security level known to the machine, but also a password, certain project associations and, though often physically or logically associated with the data, certain access privileges to data segments -- essentially the embodiment of the "need-to-know" policy. He often, of course, has further properties such as (in Multics) a "user profile" or resource quotas that do not currently concern us. There must be a mechanism by which the user is granted these properties, some presumably by himself (passwords), some by the "system security officer" (security level), and some by his "project officer" (project association). After this, each user must have available the mechanisms of security, i.e., the ability to create processes, to assign them security levels, and then to direct these processes to create data structures and assign them security levels, all of course consistent with the security policy.

STATUTORY AUDITING

Up to this point, the function of audit is basically "statutory", i.e., in consonance with DoD Security Manual 5200.28-M. A log must be maintained that includes a

---

(1) Security level, here, may be interpreted to include a formal access category, as well as a "clearance". All that is needed to implement such a policy is to impose a partial ordering on the ordered pairs of security levels represented as <classification,category>.

record of each login, each creation of a classified structure, each granting of access rights to same, each transfer of ownership of same, each modification of same, each access to same, each destruction of same, and each production of classified output. Output is generally thought of as printouts, punched cards, magnetic tapes, and disks if these are distinguishable from virtual memory [9]. Perhaps, in systems in which several active user processes may share access to the same data structure, the word "output" is to be extended to data structures created (modified) by a process but for which it can no longer guarantee control or integrity. The most troublesome "output" however, might be terminal output for which the only secure accounting method may have to be premised on severe _physical_ security restrictions.

Needless to say, for many surveillance purposes, the recording of all _unsuccessful_ attempts at logging-in or data accessing and the reasons they were denied may prove more interesting than that of the successful ones required by DoD directive. In all probability, however, the unsuccessful attempts that represent malicious (and hence more interesting) behavior will be cluttered by entries caused by incompetence, curiosity, or skeptical users. Thus tools must be developed to aid in the recognition of malicious behavior.

FUNCTIONAL AUDITING

As we mentioned in SECTION I, we foresee that auditing will serve multi-security-level computing systems not simply by keeping records of the legitimacy of their operation but also by extending their capabilities and services. These extensions derive basically from circumstances in which the judgement of the authorized users and the system's security officer concur that it would be appropriate to function within an established exceptional protocol that permits deviation from the normal validated security provisions.

One such circumstance might arise from the occurrence of a bug in an unclassified utility program that cannot be caused to manifest itself except when the program is operating on classified data. It is not difficult to build scenarios in which the requirement that the final debugged version stored as part of the system software never had the opportunity to read classified data would serve to set up such tortuous sequences of logins and logouts as to impede the effective employment of the interactive debugging facilities provided by the on-line development. The problem becomes more severe should, as might happen in the case of system software provided by the computer manufacturer, there be no personnel with clearance at the security level of the offending data sufficiently conversant with the code to debug the program efficaciously.

Similar operational problems might arise in the sanitization of documents. We are referring here to the process of extracting unclassified sections from a classified document in order to prepare a version for wider distribution. An example of sanitization has been considered by Stork [10] in a project that looked at the operations/intelligence interface. Downgrading poses no problem should the original document be composed of several segments (files) with no classified segment (file) containing unclassified data we wish to include in the sanitized version. However, such a clean segmentation of the data presupposes much foresight and care on the part of the author and supportive software on the part of computer facility.

All these problems come to a head in the case that we wish to downgrade information, i.e., to take information that had once been assigned a high security level and now, because of changed circumstances, reassign it a new, lowered security level.

For this downgrading to happen, obviously both an author and the security officer must assume responsibility. In the ordinary downgrading or sanitization of paper documents, the author must read the document carefully so as to take responsibility for the security level of its contents. Naturally, the same would be required for the release of data contained in a multi-security-level computer system. We would certainly require the author to obtain a printout of the document in question and to inspect it carefully. But there is a limit to what can be expected of any human. Can we hold him responsible for every non-printable character that could not show up in that printout, every least significant digit in every table, for the number of spaces separating each pair of sentences? Obviously not. Yet, each of these is a possible communication channel by which a Trojan horse could take advantage of the reclassification event to transmit data from classified segments to unclassified ones.

In order to make these mechanisms for exceptional operations available, we must support them with a variety of audit software tools. It is this support that we are referring to as "functional auditing". At the least, we would expect trusted programs that could perform bit-by-bit comparisons of "before and after" segments or portions of segments and which can preserve before and after segment versions for investigative purposes, should any doubts arise in the future.

To be more specific, the following sequence of events is likely in the movement of information from one security level to a lower security level. First the owner (or author) of the document composes the information to be downgraded. A request is then made external to the computer system (phone, letter, etc.) for specific downgrading action by the System Security Office (SSO). Both the SSO and author would then compare the document before and after downgrading to insure that the "after" document was the same as the before.

Thus, by providing a set of audit-like functions, it may be possible to facilitate the user interface and to allow for the sanitization and downgrading of information.

PROTECTING AGAINST THE TRUSTED USER

The DoD policy for personnel security specifically addresses the establishment of levels of security for an individual (by "measuring" the individual's trustworthiness). In the event that a user breaches his trust, the audit log provides individual accountability for access to information. Such protection against the trusted user is even more significant in non-DoD systems where policies for placing trust are not well established and the trusted user becomes a more serious threat. It should be noted, however, that as audit is used more as a part of the security mechanism, more effective tools for dealing with the audit log must be developed.

CONTINUAL HARDWARE AND SOFTWARE MONITORING

A security kernel is dependent on both hardware and software for its complete mediation of all data references. Clearly, no system, no matter how complete the verification of its programs, can be guaranteed secure in the face of hardware failures that effect its very protection mechanisms. It is therefore necessary to have present a program that continually exercises the hardware elements that support the security control mechanisms. This program is often referred to as "the subverter" [11]. The subverter is a hardware surveillance mechanism that provides protection against the probabilistic failure of the hardware. The subverter can be invoked at regular intervals to provide any specific value for the mean time between security failure (2).Incidentally, it is necessary to have a mechanism to turn off the audit function for this subverter program. Otherwise, our suspicious behavior log would fast be saturated.

While the subverter program is an obvious necessity, it is not entirely adequate as a protection against hardware-based security compromise. Because it cannot exhaustively test the hardware, the subverter may not find a well placed trapdoor, although it may find an accidentally placed one (see Karger and Schell's [12] experience with Multics).

Another possibility for "de-validation" of a verified system is that the proof of security may depend not only on hardware reliability but also on the correctness of certain actions on the part of the operator, e.g., the setting of switches or the typing of parameters when initializing the system. It would be best if this were not the case. If it proves impossible to avoid, redundant controls could minimize the danger and the parameter settings could be entered in the audit log.

----

(2) A draft Honeywell study [3] attempts a rough estimate of these probabilities. For a specific system measured, the mean time between failure is 7100 hours and the mean time between security failure is 1,920,000 hours.

DETECTION OF PENETRATION ATTEMPTS

One way often suggested to safeguard data is to try to catch the would-be penetrators in _flagrante delicto_. The advantage of being able to do this in the case that the system has not been fully validated is self-evident. There remain, though, advantages even when the system has been demonstrated secure, i.e., even when we know that the penetration attempt is doomed to failure. For one thing, we are catching an individual who, if not caught, might go on to penetrate another, more vulnerable system. Secondly, a validated system is, in our meaning, simply one whose security (at least in the sense of satisfying the *-property) has been proved. This does not imply a general "correctness of program", i.e., that the system has all good properties. So the penetrator, though he may not be able to gain access to data he is not entitled to, may succeed in attacking the robustness of the system, or he may be able to deny services to others.

Password Guessing

Probably more attention has been paid to password guessing than any other form of surveillance [13]. There are perhaps two reasons for this. The first is that accountability mechanisms fail drastically if it is possible to impersonate beyond distinction a more privileged user. This need not be true. There could be nested passwords (or even other, more complicated, behavioral prerequisites) demanded before the granting of deeper privileges. In a theoretical sense, this only changes the probability of good guessing. But, assuming the system storage of these "keys" is itself secure, such a system would certainly provide practical protection against serious impersonation attempts except those, of course, based on information yielded by the privileged user.

The other main reason for concentrating on the detection of password guessing as a surveillance goal is that, in most systems, the penetrator has, before he logs on, only the weakest resources available. He cannot, for example, run a machine language program designed to subvert both the surveillance of his operations and the security of certain sensitive data. He must deal with the system at the highest, system command level and therefore where it is best equipped to protect itself.

Our feeling is that the auditing of all incorrect passwords and simple protective routines, e.g., temporary withdrawal of all privileges to any user whose password appears to be under attack (though frought with integrity problems) certainly makes sense but is not likely to catch the dangerous fish.

Pattern Matching

One of the deepest problems in security preservation on a multi-security-level computer system is the possibility that a Trojan horse might be able to transmit highly classified information to a "listener" program of low security clearance by channels which do not represent formal violations of the *-property. That is, though the system has been validated in the sense that it has been proved the "listener" cannot gain read access to the data, it may be possible for the Trojan horse to communicate through what are known as covert channels [14]. Practical steps can be taken to minimize such a danger. One could, for example, respond to all I/O requests in a uniform, "maximum" time and maintain separate process time queues, if not for each user, then for each security classification. There are two problems inherent in such an approach. The first is that the penalty in system performance associated with such a strategy, unlike that (currently estimated as a few percent) associated with *-property satisfaction, might be quite unbearable. The second problem is that the criterion we are asking, viz, that no such channel remains, is rather less precise than the extremely simple *-property and may prove a great deal more difficult to demonstrate formally.

One response sometimes made to this problem is that any such Trojan horse must set up a peculiar behavior pattern that could, perhaps, be detected by the surveillance apparatus. Perhaps. The use of software surveillance techniques is discussed in a later section.

ACTIVE DEFENSES

It has been suggested that, associated with the surveillance system, there could be more active, entrapment procedures whose goal is, by apparently yielding to the penetrator, to catch him and/or to lead him into innocuous pathways. Some study of this possibility has been pursued by D. Hollingsworth [15] of the

Rand Corporation. We shall discuss this idea at greater length in Section IV.

ASSESSMENT OF PRIOR DAMAGE

The main function of the audit log is to serve as the electronic counterpart to the familiar security systems employed for the control of top secret information in paper form. Each copy of a classified object is to be registered in the custody of a particular individual and all transfers of custody and object destructions are to be carried out according to established procedures. Should any breach of security regulations occur, there is a clear initial point for the assignment of guilt.

However, particularly in the case of a non-validated monitor, one would hope that the system security officer could use the on-line facilities for examining the audit log for clues to possible attempts at penetrating the system. It would be *especially useful*, in case there was evidence of a successful penetration, to be able to try to paste together a picture of just which data was compromised and by what means.

One major problem may turn out to be not a lack of information but a surfeit of it. The security manual cited previously, DoD 5200.28-M, while specifying that records be kept, e.g., "each access to a classified file and the nature of the access", does contain the escape clause:

> b. The potential means by which a computer system can be adequately secured are virtually unlimited. The safeguards adopted must be consistent with available technology, the frequency of processing, the classification of the data handled or the information to be produced, the environment in which the ADP System operates, the degree of risk which can be tolerated, and other factors which may be unique to the installation involved. Rigid adherence to all techniques, methodologies, and requirements discussed in this manual could adversely impact upon the present and future use of the system under today's rapidly changing ADP technology. This technology is dynamic and the methods chosen to secure a particular system must accommodate new developments without degrading the level of protection.

c. The techniques, methodologies, and procedures in this manual, however, represent an approved method of securing a remotely accessed resource-sharing computer system in a multi-level security mode as prescribed by DoD Directive 5200.28, "Security Requirements for Automatic Data Processing (ADP) Systems", December 18, 1972. It is understood that all of the techniques described in this manual may not be economically justified after a cost versus risk evaluation. Therefore, selected subsets of the techniques included in this manual, with appropriate trade-offs, may be used to gain the level of security required for classification, category, etc., to be secured. In addition, techniques not necessarily included in this manual may be used so long as such methods provide the degree of security specified in DoD Directive 5200.28.

We have already mentioned that it would be necessary to exempt the attempts at prohibited data accesses by the "subverter" from being entered in the audit log. There will no doubt be other exceptions, but we would still expect the cumulative contents of the audit log to take on mammoth proportions. As a result we feel that the system security officer must be provided on-line tools, not just to set flags that determine which classes of events are logged, but also convenient tools for the selection, editing, and correlation of that data which is preserved. These tools should include a small language allowing him to form, at the least, boolean combinations of conditions, somewhat similar to the facilities provided by conventional text search systems.

Incidentally, the audit log will not be the sole source of clues useful in damage assessment. Damaged directories might, for example, help establish a trail pointing at data whose security is suspect.

ACCIDENT PREVENTION AND DETECTION

The security officer of any installation dealing with classified material must spend almost all his time concerned with the accidental, non-malicious failure to protect classified information according to regulations. Only a very small percentage of his activities are involved with the actual disclosure of information, even accidental disclosure. As for purposeful espionage, most security officers must surely serve their entire tour of duty without ever once coming into contact with such an incident. We should, in general, expect that accidental infractions would also predominate for multi-security level computer systems. The occurrence of willful pene-tration attempts are higher in industrial and, especially, commercial applications where they often represent monetary theft [16]. They may also be somewhat higher than we have been accustomed to in those military computer applications that present sufficiently tempting concentrations of information. Nevertheless, accidental infractions will always predominate and each must be treated as a possible disclosure and a certain nuisance.

One of the most practical functions of the system security procedures, including those of audit and surveil-lance, is to minimize the occurrence of such infractions and to correctly calculate the consequences of any that do occur. The software procedures should, in fact, be much more effective than the control of printed reports since each data structure is, unless converted to "output", under the continual control of the programs responsible for security. That is, rather than fearing the security problems associated with multi-security level computer systems, we should perhaps realize that truly validated systems are much safer repositories of classified data than those based on the normal procedures for handling such data in paper form.

Software bugs can lead to attempts at illegal memory references. The combination of a validated kernel and a program to continually check the hardware protection mech-anisms ensures that any such accidental reference cannot access data to which the process is not entitled. In unverified systems, some of this burden must be passed to the audit system which would, at least, record any such access. Another important use of the audit system is to record any accidental disclosures of information stemming from conscious decisions to relax the *-property, e.g., those discussed above in reference to either the debugging of unverified software operating on classified data bases or the downgrading of classified material.

SECTION III

CONSTRAINTS

Since we hope this report might serve as a partial checklist in the contracting and design of the audit and surveillance apparatus of multi-level systems, it is imperative that we discuss not just the purposes of audit and surveillance and the possible strategies for implementing them but also the constraints that may function to restrict those strategies.

PRIVACY

In an abstract sense, privacy and security are the same problem. Our methods for the mathematical modeling, specification, and validation of security kernels are sufficiently general to enforce many privacy policies. Security levels define a partial ordering for each type of access (read, read-write, execute, etc.) and everything that we have done abstractly is dedicated to the integrity of that partial ordering. Our methodology would remain valid were we to employ a different (privacy instead of security) policy to define that ordering.

However, from a practical point of view, almost everything we have been saying about the audit and surveillance assumes that in our instantiation of this model, there will always be a maximal element with access to all on-line information. That element, of course, is called "the system security officer". While we seem quite dependent on this "super-maximal" element, its existence appears in contradiction to any enacted or proposed civilian privacy policies [17].

The existence of a maximal element presents no problem as long as we are concerned with purely military systems operating completely within DoD regulations. A problem would only arise if the Defense Department were to find advantages in sharing a multi-level secure system with uncleared civilian users, e.g., those at a university. Under such circumstances, the non-existence of the system security officer (or, in more realistic terms, a redefinition of his prerogatives), may not be the only constraint placed on our security system. In particular, certain portions of the audit log and surveillance capabilities may become prohibited in themselves, independent of who may access them.

PERFORMANCE

Enough evidence has been derived from the AFDSC Multics -- its security mechanisms being sufficiently similar to that of a true kernel -- to know that basic monitoring of all memory references will probably impose an inconsequential loss in performance, perhaps less than 1% in system response time. Such a statement must assume a virtual environment in which we are already paying the expense of virtual addressing, but that would seem a fair assumption as long as we are speaking of future computer utilities. Most systems keep a log of some transactions. Almost all, for example, record logins and logouts and requests for outputs as well as cumulative figures related to the charging algorithms, e.g., CPU time, I-O usage, etc. But, what we believe to be the first system capable of recording all unqualified attempts at segment access, that of the Air Force Data Services Center, is only now becoming operational. We have no evidence as to the toll the requirement for auditing all segment accesses will take of the system performance. We would not be surprised to learn that it proved necessary to relax some of these requirements. Finally, all surveillance techniques, e.g., "pattern matching" or "entrapment," must also be subjected to analyses in terms of their impacts on system performance.

COST EFFECTIVENESS

The expense of any suggested audit or surveillance system must be weighed against its benefits. Of course, any of the items discussed above which adversely affect performance must also detract from the system's cost-effectiveness. But we are concerned here more with fixed costs. These must include the design and implementation of the audit software. For certifiable systems, we must add the cost of verification and for non-certifiable systems, that of testing for any parts of the audit mechanism that must be trustworthy.

In addition, we must be concerned with installation costs. The keeping of very extensive audit logs may require an extra disc drive and the long-term storage, both extra tape drives, and an enormous tape library. Furthermore, extra computer room personnel may be needed for maintaining these records and the office of the computer security officer may have to be expanded to provide assistance in the surveillance and evaluation of all this data.

INTEGRITY

It is possible that there are some conflicts between integrity and security. For example, the desire that no person or event may, whether accidentally or purposefully, destroy important data or programs argues for the existence of multiple copies, even in physically separated locations. Security considerations would, to the contrary, argue for a single copy, limited to on-line storage where it is always subject to our verifiable protection mechanisms. It is not so easy to think of a conflict between integrity and audit or surveillance considerations, but we are including "integrity" in our checklist for fear we have not thought it out sufficiently.

FLEXIBILITY

What happens to a validated system when you need to modify it? Our techniques for demonstrating the "security-correctness" of a kernel are predicated on a hierarchical design technique to be exploited for purposes both of specification and verification. This hierarchical structure would also be an aid should any modification of the system become necessary. It is hard to say how likely such changes are. On the one hand, the difficulties in proving a system correct might tend to motivate the kernel's being well-defined up to, perhaps, the setting of system parameters. This would imply a trade-off (at least until verification techniques become less burdensome) in favor of less flexibility and, consequently, we might well expect future changes. Each such change would carry the burden of a new proof of correctness, supported hopefully by the invariance of various hierarchical layers in the previous proof. Nonetheless, it would represent a burden. On the other hand, to the degree we manage to bound the functions of the kernel, the more likely will the changes that occur be external to the verified code.

What does this have to do with audit and surveillance? First and foremost, some parts of these systems must be within the kernel. This is implied by these systems having the right and duty to monitor attempts to access <u>every</u> segment. Such partial inclusion within the kernel implies flexibility on the design of the audit mechanisms.

With regard to the portions of the audit and surveillance systems outside the kernel, the impact of this problem is probably little more than the observation that it would pay to design these portions flexibly so as to be prepared to accept kernel changes gracefully.

SECTION IV

STRATEGIES

The purpose of this section is to discuss a number of means that might be employed to achieve the goals of Section II subject to the constraints of Section III.

BASIC AUDIT

Though possibly relieved by an escape clause (see page 18), Section V of DoD 5200.28-M demands that we maintain an audit log which records:

1. each login;

2. each creation of a new classified file (for MULTICS, read segment);

3. each access to a classified file (for Multics, read initiation of a classified segment) and its nature; and

4. each production of accountable classified output.

We could also add as possibilities:

5-8. each failed attempt at 1-4 above, with reason for denial;

9. each update of a file (segment);

10. each change of access privileges;

11. each change to a directory;

12. each change of audit type;

13. each attempted "ring" violation;

14. certain operator actions, e.g., changing system configuration or parameters;

15. each destruction of a classified file;

16. each deliberate *-property violation, e.g., downgrading;

17. each effective action of the system security officer;

18. each change of monitor version;

19. each alarm to the system security officer from any surveillance program;

20. each detected hardware failure;

21. each system crash;

22. statistics on user resource consumption.

Care should be taken in the basic audit of the system to identify and record the minimum amount of information. It is easy to postulate an audit system that records so much information that the timely selection of interesting data is impossible. Thus the choice of data and the tools to analyze the data are important.

The audit system comprises both the logging mechanism that records some subset of the above items and the software available to the system security officer. This software is of two types. The first provides the security officer control over the auditing process by allowing him to choose (a) which events are audited for (b) which processes/users. He must, as a minimum, be capable of turning off the monitoring of faults associated with the program that maintains a continual check on the hardware protection mechanisms. This control over the subverter program could be satisfied by the capability of assigning each process an overall audit type that decides whether all its auditable actions are recorded in the audit log. It might be more useful to associate with each process, not an audit bit, but an audit vector that decides the logging of each type of audit-worthy event independently.

The second type of audit software comprises the tools provided to the security officer for the analysis of the recorded data. This must include tools for searching the voluminous records for events satisfying security officer defined predicates and probably should include others for the correlation of such events. This software should be augmented in response to the security officer's needs as they develop in light of experience with the audit software and the event patterns of the entire system.

SURVEILLANCE

Besides more esoteric surveillance strategies such as "pattern-matching" or entrapment discussed below, the security officer should be provided some simple software that he can use to observe security-related events while they are in progress. This software might include alarm systems that call his attention to patent attempts at password guessing or to especially high frequencies of attempted protection violations, on the assumption that such high frequency events may represent malicious intent. The software should associate specific terminals and processes with the suspicious activity.

We would also expect the security officer to have, at least in those circumstances in which there is no privacy constraint, a Big Brother capability of selectively tuning in on various processes and events, watching them in progress, reading information as it is updated, noting resource requests, intercepting interprocess communication, etc. This would all represent a rather basic spying facility with little in the way of data processing aids to help refine and correlate the evidence. Such correlation aids would be part of the more sophisticated strategies discussed below.

MULTIPLE ENTRY INFORMATION

Information, whether we think of it as data, or the value of state variables, or entries in directories or access control lists, or system parameters, can be stored in several locations that can be checked for consistency. This redundant storage has some attraction as a surveillance tool for the discovery of would-be penetrators. In the case of non-certified systems, such redundancies can also provide an extra layer of protection.

Protection would come from secrecy, i.e., from the fact that attempts to penetrate on the basis of experiment and deduction, in ignorance of a complete system description, would be very unlikely to put the pieces together. This unlikelihood can be arbitrarily increased by imposing not a simple repetition of data but by the imposition of an arbitrarily complex relation among the various data elements. Particularly sensitive information may be protected, even against spies with a knowledge of the system, by making this "key" relationship subject to privileged updating.

An auxiliary advantage of such redundancy is the protection it lends the integrity of system against errors, particularly on the part of the operator and the system security officer.

Beyond the use of such redundancy for the protection of system (and user) integrity and for additional security protection in non-certified systems, the imposition of calculable relations among information items can serve as an important audit and surveillance tool. All actions that are denied because of the inconsistency of such data can be recorded in the audit log just like any other attempt at a protection violation. Furthermore, a system program could move through the active processes making spot checks on the consistency of these items even when that consistency is not requirec by the user for the exercise of particular privileges. After making some allowance for the fact that the redundant items cannot be changed simultaneously, the discrepancies could be brought to the attention of the security officer.

PATTERN RECOGNITION AND STATISTICAL STUDIES

One of the techniques often suggested for the surveillance of multi-level secure systems is the real-time search for suspicious patterns of behavior on the part of particular processes. There is, first of all, the superficial behavior patterns associated with password guessing or attempts to access prohibited directories or access control lists. Another target of these proposed pattern matching techniques are the more sophisticated attempts at the theft of classified data by "Morse Code". The situation we fear is that a "Trojan horse", e.g., an untrusted utility program, being used to process classified data could manage to "telegraph" classified information to an unclassified (or lower classified) "listener" program. That is, the Trojan horse might try to set up a pattern of behavior, e.g., a time-sequence of resource (virtual memory, I-O facilities, etc.) utilization or saturation that could be monitored, admittedly in the presence of noise, by the listener program. The reception process might, for example, be based upon denials of listener requests for the same resources. In a recent informal experiment at MIT, the Multics paging mechanism was employed to telegraph a message at the rate of about one character per second with about a bit error rate of 108-29. In some more "paranoid" scenarios, the Trojan horse is imbedded in the hardware.

The hope expressed by advocates of pattern-matching surveillance techniques is that, since the telegraphed message is imposing a patterned behavior on an otherwise random sequence of events, that same fact might be exploited for the detection of the Trojan horse. To our knowledge, nothing has really been done to substantiate this possibility. The fact that, as viewed by the system, normal resource requests arrive randomly hardly implies that their arrivals, especially -- and this is the question -- their arrivals from a particular benign process, are uncorrelated. In order to send a message, the event spectrum of the Trojan horse must be different than that of normal system use. But it would always be possible to build a counter-counter response to spectral surveillance by arbitrarily lowering the bandwidth, i.e., making the telegraph channel appear more like benign behavior at a cost of lower information transmission rates. We must further keep in mind that the surveillance program would normally be trying to detect the Trojan horse among all active classified processes in complete ignorance of the techniques and code employed or even the Trojan horse's existence.

We would like to suggest that the most reasoned approached to the question of pattern matching surveillance methods is to gain a better understanding of the possibilities through the collectior of event statistics and the performance of simple experiments on systems such as Multics.

The objects on which we might wish to collect statistics fall into two classes, depending on whether we are trying to detect an external lock-picking attempt at system penetration or an internal Trojan horse telegraphy of information to a listener. In the first case, we expect the intruder to be revealed by the illegality of his moves. The objects we would be interested in would include elementary events such as illegal instructions (particularly attempted ring violations in Multics) and illegal addresses. These latter would primarily be due to segment violations, e.g., requests for non-existent segments, or segments to which the user is prohibited access (particularly to system directory or tabular information) or to which he has access privileges, but of a different type. All of these statistics should be kept in rather fine detail. System tables are to be distinguished from other protected segments, ring violations are to be classified by ring, segment denials by reason of security level are to be distinguished from those due to need-to--know, i.e., discretionary, privileges. Similar statistics would also be kept for non-virtual entities, e.g., denials of off-line printouts.

Besides these elementary events, we would also record statistics on some compound events that appear indivisible to system users. These would include all logon failures whether caused by non-authorized personal identification, invalid project or group association, incorrect password, a request for too high a security level or an attempt either to log on at a personally valid security level from a terminal which is not physically secured at that level or an attempt to logon at any level on a terminal whose physical security exceeds the user's clearance. The last represents a breach of physical security and must invoke an immediate alarm.

Much of the above material will be recorded in the audit log in any case. We are only suggesting that a little statistical data processing might help us determine if any proposed surveillance scheme has much chance of detecting external penetration threats.

With regard to the internal, Trojan horse, threats, on the other hand, it would seem necessary to carefully design specific experiments. Unlike the items mentioned above which are (though most frequently by accident rather than design) patent attempts to transgress the system's security boundaries, the Trojan horse is employing normal system components in apparently normal modes and never violates any formal protection mechanisms. We have, in fact, very little to look for, just unlikely patterns in the demands placed on deep system resources, e.g., the paging mechanism or the I-O channels or devices.

For both the internal and external threats, the statistics collected for each of the listed items must include its mean (i.e., the volume of transactions), its variance (i.e., the peculiarity of special events), and certain correlations. These would include correlations between different events, between events and processes, and between events and users. This list is needed for both the discounting of personal idiosyncrasies and for the association of a Trojan horse with a listener.

In the case of the internal Trojan horse threat, we must also collect the critical data on the frequency spectra of calls and saturation of the system components. Furthermore, we must investigate what sort of time changes are to be expected in these spectra, since such events, being normally so dependent on system load, could hardly constitute stationary processes.

Any potential criteria for the discovery of a Trojan horse that might emerge from these studies should be tested experimentally against Trojan horses we have foaled ourselves. These tests should be run, presumably on a Multics system, under both heavy and light loads by parties blind to the presence or absence of the Trojan horse. Furthermore, these tests must be designed to reveal clearly their dependence or independence of both the actual techniques employed by particular Trojan horses and the knowledge of these techniques.

ACTIVE DEFENSES

As previously referenced, Hollingsworth advocates the use of active defenses, i.e., pseudo-flaws combined with entrapment tactics, designed to lead a would-be penetrator away from system weaknesses into a path on which he seems to be gaining prohibited accesses, but is, in fact, being misled and monitored.

At the system command or text-editor level, the scenario would have the user guessing passwords, being rejected several times, then finally being accepted. The user then tries to read privileged data and, while apparently succeeding, is, in fact, fed false data from prepared files. With respect to lower level attempts at penetration, e.g., prohibited memory references generated by user defined assembly language routines, the strategy is for the illegal instruction to trap to a software routine which sets up a pattern of misleading behavior rather than the normal program abort. From the moment his behavior becomes so suspicious as to invoke the entrapment process, the user would not be only subject to the normal audit and surveillance procedures, but would also be monitored so that an alarm would be sounded and a real-time transcript of the entire user-machine conversation preserved.

One must be skeptical. All concern about password guessing, whether in reference to entrapment or pattern matching surveillance techniques, must raise the question as to whether such tactics constitute any substantial threat. Since most systems give the user no, or practically no, resources until logged on, he can do little but guess wildly. Our real threats are from much more sophisticated users who hope, in fact, to employ the extensive on-line system facilities, turning them to their own advantage. Any sophisticated user who found he could reference prohibited segments through the sole use of such high level facilities as a text-editor would be very suspicious indeed.

Still, if one could provide convincing active defenses, the advantages would be quite attractive. Hollingsworth advocates entrapment as an additional defensive mechanism for non-certifiable systems. But, catching a thief is interesting in itself, and catching a spy much more valuable. So there would be strong motives to add such facilities to certified systems also. Active defenses have the further attraction that they not only trap the culprit in the act, but they also go a long way towards establishing his intent. One must be extremely skeptical, however, about the machine's ability to understand the user's motives enough to convincingly entrap him when the total evidence of his intentions is imbedded in an arbitrary assembly language program.

SECTION V

RECOMMENDATIONS

The discussion to this point indicates that, in general, the audit of multi-security-level systems is fairly well understood as is hardware surveillance. The proposed approaches to software surveillance, at least the more imaginative ones, are on much less solid foundation. We should like in this section to suggest guidelines with respect to the study and implementation of audit and surveillance related subsystems that seem to us natural and reasonable under these circumstances.

ENGINEERING APPROACHES

We refer here to those tasks that are obligatory and sufficiently understood so that intelligent design is possible today.

Basic Auditing

ESD is involved in the construction of a number of multi-level secure systems which require audit facilities. These would include the Air Force Data Services Center [2], the Security Kernel Based Multics System, the Secure Front-End Processor to Multics, and the PDP-11/45. The audit logs required by DoD 5200.28-M must include items 1-4 of our list of 21 items at the beginning of Section IV. Though the practicality of recording all accesses to classified segments is extremely questionable, the facilities to satisfy all four requirements will exist in the two MULTICS systems just referred to. The syserr log of the AF Data Service Center will also contain records of denied accesses and illegal procedures (our items 7 and 13). Most of the other items on our list would also seem well-advised and easy to implement.

In addition to the actual logs, the system security officer must have available the relevant software described in Section IV for the control of the auditing process (the Data Services Center specification referenced above describes "audit select flags" which are equivalent to our "audit vector") and for the on-line analysis of collected data. This list includes the audit search specification language and correlation tools. Efforts should be concentrated in identifying and recording only the minimum necessary data to keep the data reduction/analysis problem from becoming too complex.

Surveillance

The "subverter", the hardware surveillance mechanism, should be included to provide protection against probabilistic hardware failure. The operation of a subverter is discussed by Karger and Schell [12]. The provision of the alarm and Big Brother systems mentioned in Section IV represent an additional surveillance capability that might be a goal for future development.

DEVELOPMENTAL APPROACHES

The main problem is the evaluation and design of the more sophisticated surveillance techniques. We would like to suggest that the steps below be taken sequentially.

Statistical Studies

In Section IV, we outlined at some length a proposed statistical study of security related events. We feel such a study should be undertaken on some selected Multics system(s). It should first be determined which statistics are currently available from extant Multics systems, e.g., MIT, Honeywell, RADC, AFDSC, and then this information should be gathered.

Incidentally, during this study statistics should be kept, to the extent possible, on the cost in machine time and (implicitly) capital equipment required for monitoring different events. This, of course, would be for the purpose of gaining more precise understanding of the performance and cost related constraints of Section III.

The main point of collecting and interpreting the statistics on security related events is to have them available as analytical aids in the feasibility study of the more sophisticated surveillance techniques, especially the pattern matching approaches to Trojan horse detection. Trojan horse detection is the most important surveillance problem because:

1. The Trojan horse remains, together, perhaps, with the possibility of espionage by cleared personnel, as one of the outstanding problems in complete computer security. This is a consequence of the fact that Morse code need not represent a *-property violation, even in certified systems.

2. We really do not know how feasible or effective pattern matching techniques are likely to be as a means of Trojan horse detection.

3. No other defense against this threat has, to our knowledge, been suggested other than the division of the machine's resources to the extent that processes of different classification share nothing. This latter approach would, however, certainly defeat many of the motives that led to the sharing of the machine in the first place.

Approach Analysis

Partly based on these statistical analyses, the various advanced surveillance techniques, e.g., pattern matching, active defenses, multiple entry of key information and more sophisticated data relations, should be subjected to serious analyses to estimate their effectiveness against both external, high level penetration attempts and internal, low level ones. In addition, the solutions must be analyzed with regard to their interaction with the constraints of Section III. In particular, the redundant storage of information described as the multiple entry approaches must be studied to establish whether there is a middle ground between a solution so complicated that it serves as an impediment to the normal use of the system and one so simple that its effect is negated by the "underground" circulation of a few macros which, in effect, convert the multiple entry system into a single entry one.

Design and Implementation

We are assuming the basic audit and surveillance mechanisms are to be implemented early. As for the more advanced surveillance techniques, we are suggesting that these be postponed until the completion of the above studies.

REFERENCES

1. Department of Defense, "Security Requirements for Automatic Data Processing (ADP) Systems," Department of Defense Manual 5200.28, December 1972.

2. J.C. Whitmore, A. Bensoussan, P.A. Green, A.M. Kobziar, and J.A. Stern, "Design for Multics Security Enhancements," ESD-TR-74-176, Honeywell Information Systems, 1974.

3. Honeywell Information Systems, "Secure Communications Processor Architecture Study," Draft, December 1975.

4. W.L. Schiller, "The Design and Specification of a Security Kernel for the PDP-11/45," ESD-TR-75-69, Electronic Systems Division, AFSC, Hanscom AF Base, Mass., May 1975.

5. J.P. Anderson, "Computer Security Technology Planning Study," ESD-TR-73-51, Volume I, James P. Anderson & Co., Fort Washington, Pennsylvania, October 1972.

6. J.P. Anderson, "Multics Evaluation," James P. Anderson and Co., ESD-TR-73-276, October 1973 (AD 777593).

7. D. Bransted, "Privacy and Protection in Operating Systems," Computer, Volume 6, Number 1, January 1973, pp. 43-47.

8. D.E. Bell and L.J. LaPadula, "Computer Security Model: Unified Exposition and Multics Interpretation," ESD-TR-75-306, Electronic Systems Division, AFSC, Hanscom AF Base, Mass., January 1976.

9. E.L. Burke, "Concept of Operation for Handling I/O in a Secure Computer at the Air Force Data Services Center (AFDSC)," ESD-TR-74-113, Electronic Systems Division, AFSC, Hanscom AF Base, Mass., April 1974.

10. D.F. Stork, "Downgrading in a Secure Multilevel Computer System: The Formulary Concept," ESD-TR-75-62, Electronic Systems Division, AFSC, Hanscom AF Base, Mass., May 1975.

11. C. Weissman, "Security Controls in the ADEPT-50 Time-Sharing System," Proceedings AFIPS 1969 FJCC, AFIPS Press, Montvale, New Jersey, 1969, pp. 119-133.

12. P.A. Karger and R.R. Schell, "Multics Security Evaluation: Vulnerability Analysis," ESD-TR-74-193, Volume II, Electronic Systems Division (AFSC), L.G. Hanscom Field, Bedford, Massachusetts, June 1974.

13. M. Gasser, "A Random Word Generator," ESD-TR-75-97, Electronic Systems Division, AFSC, Hanscom AF Base, Mass., November 1975.

14. S.B. Lipner, "Comment on the Confinement Problem," MTP-167, The MITRE Corporation, Bedford, Massachusetts, December 1975.

15. Dennis Hollingsworth, "Enhancing Computer System Security," P-5064, The Rand Corporation, Santa Monica, CA., August 1973.

16. Donn B. Parker, Susan Nycum, and S. Stephen Ohura, "Computer Abuse," Stanford Research Institute, Menlo Park, CA. November 1973.

17. Leo J. Rotenberg, "Making Computers Keep Secrets," MAC-TR-115, MIT, Cambridge, MA., February 1974.